**Proposed changes in response to reviewers' comments**
**Paper**: Modeling The Temporally Constrained Preemptions of Transient Cloud VMs
**Authors**: J.C.S Kadupitiya, Vikram Jadhao, Prateek Sharma

We thank all the reviewers for their detailed comments and insightful feedback. Most of our changes are additions to the paper text that address reviewer comments. We have grouped the comments by reviewer, and have included the summary and location of the proposed changes.

# 1 TPC meeting

1. *Relevant new EC2 published, in particular the work of Ristov et al., CLOUD'18.*

   **Response**: We will add and compare with more related work in Section 7. We will emphasize that even newer work on EC2 modeling is focused on spot price modeling which is fundamentally different to our work, and is not applicable in other transient computing environments.

   **Revision Made**: Added 3 new references and discuss the EC2 spot pricing change in related work section.

2. *Re. experiments, both the setup, configuration, and how the results are presented should be clarified. A re-analysis, with various thresholds, seems to be both valuable and covering the "hypothetical heuristics". For example, present the results by dropping jobs that are known to exceed the 24 hour termination restriction (e.g., 6 hr jobs started after 18 hrs), etc.*

   **Response**: We will fix the various evaluation exposition issues, which we individually address in the rest of this revision plan.

   **Revision Made**: We have made the evaluation exposition more precise in many places. We have also removed a few result summaries, since we found it difficult to make the summaries both complete and succinct.

# 2 5A

1. *It should be made clear that with VM-reuse, the performance impact of one preemption is as low as 3%. Because if a job experiences multiple preemptions, it will have more than 3%-5% increase in running time.*

   **Response**: We will clarify that 3% is the *expected* increase in running time, which includes the (low) probability of preemption.

   **Revision Made**: We have clarified that 3% is the *expected* increase in running time, which includes the (low) probability of preemption, in Section 6.3.

# 3 5B

1. *3.1 Is the following statement true: "In general, the preemption dynamics of a VM are determined by the supply and demand of VMs of that particular type" The authors do not provide a citation for this statement .*

   **Response**: We will add relevant references for the above claim (ExoSphere and other follow-up work).

   **Revision Made**: We have removed this line since it was not relevant to constrained preemptions studied in this paper.

2. *3.1 What timezone is used in observation 5? I assume central to match the region. But this should be clarified. The model might be better termed a termination, preemption, or revocation model*

   **Response**: Timezone local to the region being analyzed. Will clarify in Section 3.

   **Revision Made**: Added footnote saying that the time is local to the region.

3. *4.1 and 4.2 For many applications it is difficult to predict run time (especially on different cloud instances with unknown background load). Even when using the same application, different parameters can significantly change compute time. It's unclear what assumptions are made here that workloads need to be predictable.*

   **Response**: We will clarify this weak assumption in Sections 4.2 and 4.3—none of our policies require accurate job running time estimates. We will also briefly describe scenarios in which estimates are easy to obtain, such as

parameter sweeps in scientific simulations. For the job scheduling policy in Section 4.2, since we only *compare* the benefits of VM reuse, we need only rough estimates of job running times, and exact times are not required. Similarly, the checkpointing policy is not affected if the job finishes before the estimated time. If the job takes more time, the checkpointing schedule "rolls around" and again provides the same checkpointing overhead.

**Revision Made**: We have added text in Section 4.1 about the need for only rough estimates of job running time for the scheduling policy.

4. *Comparison against basic heuristics would be interesting and perhaps more comparable than EC2 algorithms.*

   **Response**: Even basic heuristics would need to be based on sound principled empirical models, which we provide. We will discuss this in Section 8.

   **Revision Made**: Further, one may consider the use of such models (heurestics) as preferred for their simplicity. However our analytical predictive model enables for a cleaner integration with the resource management service. The preemption model function also provides a measure to distill the contributions of effects ignored in its derivation (multiple preemptions, non-differentiable features) in changing the VM expected lifetimes and checkpointing policies. On the other hand, our analytical model for the Google preemptible instances derived using a principled approach informed by empirical data and based on well-defined assumptions can guide the development of simpler heuristics and modeling approaches, as well as interpretion of their results.

5. *In Fig 7 what is the distribution of job lengths? Is the difference again due to start times at the end of the 24 hour period?*

   **Response**: We will clarify that figure 7 uses uniformly distributed job lengths.

   **Revision Made**: We have clarified that figure 7 uses uniformly distributed job lengths, in Section 6.1.

6. *In Fig 8 the the authors simply picked a job and checkpoint length.*

   **Revision Made**: We will clarify that Fig 8 evaluates a wide range of job lengths [0–9 hours], in Section 6.2.2.

7. *6.3 TFig 9 The authors should describe thee characteristics and it would likely be more interesting to quite different applications.*

   **Response**: We will add details about the jobs and their running time characteristics.

   **Revision Made**: We have added details about the jobs and their running time characteristics at the start of the eval section.

# 4 5C

1. *Assumptions that require references or reformulation: On page 6, the authors assume that multiple preemptions are not worth modeling. They assume "most transient computing systems seek to avoid repeated preemptions". Is there evidence for this?*

   **Response**: Prior work, including this paper, temporarily blacklists VMs with high preemption rates. We will cite the related work. We will emphasize on page 6 that while multiple preemptions are certainly worth modeling, we wanted to provide a model for the common case (one preemption) as part of this initial work on constrained preemption modeling.

   **Revision Made**: We have clarified in Section 4.1 that multiple preemptions have a low probability, which is why our initial model does not try to estimate it.

2. *Assumptions that require references or reformulation: On page 8, the authors assume most batch jobs are comprised of tasks with little variability in runtime and execution characteristics. Is there evidence for this?*

   **Response**: We make this assumption only for ease of evaluation and minimizing stochasticity to allow us to easily compare different system configurations. We will clarify this in the evaluation section.

   **Revision Made**: We have clarified this in Section 4.2.

3. *Claims that require more careful analysis or reformulation: CRITICAL: The most important issue with this article is the way the results, and especially the experimental results, are presented. The authors claim improvements of "X" or, less frequently, "up to X", but do not position carefully this result with minima or other distributional information. This results in formulation that, while consistent with the results, does not fully characterize the results and is misleading. (This could be fixed with shepherding.)*

4. *Claims that require more careful analysis or reformulation: Related to the point above, but less critical, the authors should try to include specific numbers rather than qualitative elements in their results. For example, in the Result in Section 6.1, the terms "significantly" and "slightly" could be avoided.*

   **Response**: We will go over all results and present ranges wherever required, and be more precise when summarizing the results, especially when explaining Figure 5, 6.

   **Revision Made**: Made clarification changes in the entire eval section when explaining results.

5. *Experimental setup that requires a better explanation: CRITICAL: The experimental results in Section 6 are based on a setup that does not describe the workload characteristics (but the authors indicate the type of applications), the failure characteristics and way of injecting them in the real-world environment, and the environment characteristics (it is part of the Google Cloud Platform). (This could be fixed with shepherding.)*

   **Response**: We will include all workload characteristics (running times and cluster sizes) in Section 6. All of the following scientific simulations ran with its defualt configurations available in the github code and the optimum workload characteristics found in the exploration phase are Nanoconfinement: 4 of n1-highcpu-16, 851.325 seconds, Shapes: 4 of n1-highcpu-16, 548.55 seconds, LULESH*50: 8 of n1-highcpu-8, 762.225 seconds.

   **Revision Made**: Added the workload characteristics. TOFIX: fail characteristics and env characteristics.

6. *Experimental setup that requires a better explanation: In Section 6.1, the impact on jobs shorter than 5 hours should be explained more carefully.*

   **Response**: We will provide more explanation of how jobs shorter than 5 hours are affected, at the start of Section 6.3. Briefly, it is fundamentally a result of bathtub initial high failure rates.

   **Revision Made**: We have added clarification for how short jobs are affected by the high initial rates of failures in Section 6.

7. *Experimental results that could be better explained: In Section 6.2, the failure probability after 18 hours is now fixed, but it seems that the approach would lead to a failure probability decreasing with time after 18 hours. The authors could comment on this situation and explain the (counter-)intuitive result.*

   **Response**: In Section 6.2, we will explain how our policy deals with the "edge case" of jobs starting near 24 hours, and how it leads to the counter intuitive result. Briefly, since we always start on a new VM after 18 hours, the job will always start at t=0, and have a failure probability of a new VM, which is constant and is equal to the left side of the curve.

   **Revision Made**: Added text describing the job failure probability near the 24 hour mark in Section 6.2.1.

# 5 5D

1. *The rational of why the failures seem to have the early high failure rate is not clear. I mean, if I have 100 VMs that have been running for 10 hours, and 100 VMs that I just started, does Google really take into account the VM lifetime and prioritize preempting young VMs? This is a very interesting question that you may be able to answer from your data set.*

   **Response**: We will summarize our observations to this question in Section 3. Briefly, yes, the number of simultaneous VMs launched has an effect on their failure rate.

   **Revision Made**: Added the clarification in Section 3 about how simultaneous launches affect preemptions and how we account for that.

2. *Please clarify your assumptions for your lifetime and makespan analysis. Most of cloud applications are either web services in which failing only effects the currently running web requests, or are batch processing using Spark (or something like) in which case, the framework is resilient to single node failures (either due to reexecuting the lineage of the task or through repeating the task). For those cloud workloads, failures are not as painful. So your scheduling and checkpointing optimizations are mainly usefull for long running science applications in the cloud. It will be good to clarify this in the paper.*

   **Response**: We will add this clarification. Yes, our current policies only target long-running jobs that arise in scientific computing, CICD pipelines, and many other areas. Our model allows designing policies for other distributed applications such as Spark. We note that even Spark jobs are affected by single node preemptions, and will add the relevant references [Flint].

**Revision Made**: Added text at start of Section 4 clarifying that we assume simple batch jobs, and more complex failure models are part of our future work.

3. *Some of the details are missing: When did you lunch the VMs? I mean, did you lunch VMs in batches every hour? Did you test with VMs with different size of memory.*

   **Response**: We will include some more details about VM launch times. The launch times can be inferred from Figure 2b.

   **Revision Made**: Added details in the methodology paragraph in Section 3.

4. *Some of the details are missing: For the evaluation in section 6. The details of the experiment are missing. When did you run the VMs? How many VMs? In which region? How many preemptions did you face during your experiment?*

   **Response**: We will add these details in Section 6. When did you run the VMs? This is answered above. How many VMs? 870 VMs. In which region? us-central1-c: 347, us-east1-b: 212, us-west1-a: 311 How many preemptions did you face during your experiment? Total VM count passed 24 hrs: 351

   **Revision Made**: Added clarification at start of Section 6 that the details are provided in Section 3.

# 6 5E

1. *Curious how much could be done with a simpler modelling approach.*

   **Response**: We discuss this in Sections 3 and 8. Briefly, simpler models would not be able to generalize across VM types, nor be usable for sophisticated policies. Our "complex" model allows extensions to policies to be developed in an easier manner.

   **Revision Made**: Added text in discussion section.

2. *24 hour limit is actually a pretty artificial limit imposed by Google. Therefore, its unclear to me how well the approach would continue to hold for a system with higher levels of demand. For instance if the authors approach is successful, then I would expect for Google to change the rules to make it less successful. The question is really how much the approach implemented by Google generalizes to other system environments. The authors don't make that case, and I'm skeptical that it can be made.*

   **Response**: We will expand our discussion of this in Section 8. Briefly, we will emphasize that beacause bathtub preemptions are good for the applications, they will continue to remain a good choice for constrained preemptions. Moreover, our modeling approach works across a wide range of instance types and is able to model CDFs of instances with both very high and very low failure rates, and thus is already general.

   **Revision Made**: Added text in discussion section.